# Two-Factor Authentication (2FA) : Part 3

Five PHP endpoints

- Five PHP end points
  - status.php
  - setup.php
  - reset.php
  - login.php
  - verify.php

## Five PHP endpoints

- status.php
- setup.php
- reset.php
- login.php
- verify.php

## status.php

- 2FA STATUS ENDPOINT
  - GET /status.php?username=john
  - POST /status.php with {"username": "john", "password": "password123"}
  - Shows current 2FA status for debugging and user information

## Request and Response

### Request

- Handle both GET and POST requests
- Sets
  - The $username
  - The $authenticated true or false

- For the GET request, we return simple information.
- For the PUT + password, we return detailed information.

```php
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    $username = isset($_GET['username']) ? $_GET['username'] : null;
    $authenticated = false; // GET requests show limited info
} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $input = json_decode(file_get_contents('php://input'), true);
    if ($input && isset($input['username'])) {
        $username = $input['username'];
        // Check password for detailed info
        if (isset($input['password'])) {
            $user = getUser($username);
            if ($user && $user['password'] === $input['password']) {
                $authenticated = true;
            }
        }
    }
}
```

## Response

- Make basic/detailed information and make a response

```
// Basic status (available to everyone)
$basic_status = [
    'username' => $username,
    'totp_enabled' => $user['totp_enabled'],
    'has_secret' => !empty($user['totp_secret'])
];
$detailed_status = [
    'username' => $username, 'user_id' => $user['id'], 'totp_enabled' => $user['totp_enabled'],
    'has_secret' => !empty($user['totp_secret']),
    'secret_length' => !empty($user['totp_secret']) ? strlen($user['totp_secret']) : 0,
    'setup_status' => [
        'ready_for_setup' => !$user['totp_enabled'] && empty($user['totp_secret']),
        'setup_in_progress' => !$user['totp_enabled'] && !empty($user['totp_secret']),
        'fully_enabled' => $user['totp_enabled'] && !empty($user['totp_secret']),
        'inconsistent_state' => $user['totp_enabled'] && empty($user['totp_secret'])
    ],
    'available_actions' => []
];
if ($authenticated) {sendJsonResponse($detailed_status);} else {sendJsonResponse($basic_status);}
```

- Fill in the $detailed_status for authorized users

```
// Determine available actions
if (!$user['totp_enabled']) {
    $detailed_status['available_actions'][] = 'setup_2fa';
} else {
    $detailed_status['available_actions'][] = 'reset_2fa';
    $detailed_status['available_actions'][] = 'login_with_2fa';
}

// Add troubleshooting info
if ($user['totp_enabled'] && empty($user['totp_secret'])) {
    $detailed_status['warning'] = 'Inconsistent state: 2FA enabled but no secret stored';
    $detailed_status['suggested_action'] = 'Reset 2FA and set up again';
}
```

## setup.php

- 2FA SETUP ENDPOINT
  - POST /setup.php
  - Body: {"username": "john", "password": "password123", "force_reset": false}

## Request and Response

### Request

```
$username = $input['username'];
$password = $input['password'];
$force_reset = isset($input['force_reset']) ? $input['force_reset'] : false;

$user = getUser($username);
```

## Response - 2FA enabled user

- When the user is 2FA enabled, we return with 400.

```
sendJsonResponse([
    'error' => '2FA is already enabled for this user',
    'current_status' => [
        'totp_enabled' => true,
        'setup_date' => 'Previously configured'
    ],
    'options' => [
        'reset_first' => 'Use POST /reset.php to reset 2FA first',
        'force_reset' => 'Add "force_reset": true to this request to reset and re-setup',
        'check_status' => 'Use POST /status.php to check current status'
    ],
    'security_warning' => 'Resetting 2FA will require new authenticator setup'
], 400);
```

## Response - 2FA Disabled user

- Generate secret and QR information.

```php
// Generate new TOTP secret
$secret = SimpleTOTP::generateSecret();

// Store secret temporarily (not yet enabled)
updateUser($username, ['totp_secret' => base64_encode($secret)]);

// Generate QR code URLs from multiple services for fallback
$qr_urls = SimpleTOTP::getQRCodeURLs($secret, $username, 'PHP 2FA Demo');

// Primary QR code URL (using QR Server API)
$primary_qr_url = SimpleTOTP::getQRCodeURL($secret, $username, 'PHP 2FA Demo');

// Generate current code for verification (demo purposes only)
$current_code = SimpleTOTP::generateCode($secret);
```

- Make a response and return

```
$response = [
    'message' => $force_reset ? 'Previous 2FA reset, new setup ready': 'Scan QR code with your authenticator app',
    'qr_code_url' => $primary_qr_url,
    'qr_code_urls' => $qr_urls,
    'manual_entry_key' => SimpleTOTP::base32Encode($secret),
    'current_code' => $current_code,  // For demo purposes only
];

if ($force_reset) {
    $response['reset_performed'] = true;
    $response['warning'] = 'Previous 2FA configuration was reset. Set up your authenticator app with the new QR code.';
}

sendJsonResponse($response);
```

## Response - 2FA Disabled user with force reset

- When the force reset flag is set, we need to update the user and response.
- This is similar to reset as it updates users.

```
// Force reset requested - reset first, then continue with setup
$resetData = [
    'totp_secret' => null,
    'totp_enabled' => false
];
updateUser($username, $resetData);

// Reload user data after reset
$user = getUser($username);
// Add more information to the $response
$response['reset_performed'] = true;
$response['warning'] = 'Previous 2FA configuration was reset. Set up your authenticator app with the new QR code.';
```

## reset.php

- 2FA RESET ENDPOINT
  - POST /reset.php
  - Body: {"username": "john", "password": "password123", "confirm_reset": true}

## Request and Response

### Request

```php
$input = json_decode(file_get_contents('php://input'), true);

if (!$input || !isset($input['username']) || !isset($input['password'])) {
    sendJsonResponse(['error' => 'Username and password required'], 400);
}

$username = $input['username'];
$password = $input['password'];
$confirm_reset = isset($input['confirm_reset']) ? $input['confirm_reset'] : false;
```

- Security check: require explicit confirmation
- If confirm_reset is not true, return an error message.
    - This is why to make the `confirm_rest` parameter in the input JSON true.

```
$user = getUser($username);

if (!$confirm_reset) {
    sendJsonResponse([
        'error' => 'Reset confirmation required',
        'message' => 'To reset 2FA, send confirm_reset: true in your request',
        'warning' => 'This will disable 2FA and require re-setup',
        'current_status' => [
            'totp_enabled' => $user['totp_enabled'],
            'has_secret' => !empty($user['totp_secret'])
        ]
    ], 400);
}
```

### Response

- Reset 2FA with the updateUser method.
- Return the response.

```
$resetData = [
    'totp_secret' => null,      // Clear the secret
    'totp_enabled' => false     // Disable 2FA
];
$success = updateUser($username, $resetData);

if ($success) {
    sendJsonResponse([
        'message' => '2FA has been successfully reset',
        'username' => $username,
        'status' => 'reset_complete',
        'next_steps' => [
            '1. 2FA is now disabled for this user',
            '2. User can now set up 2FA again using /setup.php',
            '3. Old authenticator entries should be deleted',
            '4. New QR code will be generated during setup'
        ],
        'security_note' => 'User account is now secured with password only'
    ]);
```

### test/quick_reset.php

- This is a utility application to reset users without the GUI.
- Usage: `php test/quick_reset.php ARGUMENT`.
  - --all
  - john or admin

## Reset All users

```
loadUsers();
global $users;

foreach ($users as $user => $data) {
    if ($data['totp_enabled']) {
        $resetData = ['totp_secret' => null, 'totp_enabled' => false];
        updateUser($user, $resetData);
        echo "✅ Reset 2FA for user: {$user}\n";
    } else {
        echo "ℹ️  2FA not enabled for: {$user}\n";
    }
}
```

## Reset specific user

```php
$user = getUser($username);
// Perform reset
$resetData = [
    'totp_secret' => null,
    'totp_enabled' => false
];
$success = updateUser($username, $resetData);
```

## login.php

- 2FA LOGIN ENDPOINT
  - POST /login.php
  - Body: {"username": "john", "password": "password123", "totp_code": "123456"}

## Request and Response

### Request

- Check if 2FA is enabled for this user

```
$input = json_decode(file_get_contents('php://input'), true);

$username = $input['username'];
$password = $input['password'];
$totp_code = $input['totp_code'] ?? null;
```

- Get user and check totp_enabled status

```
$user = getUser($username);

// Check if 2FA is enabled for this user
if (!$user['totp_enabled']) {
    // Login without 2FA (user hasn't set it up yet)
    sendJsonResponse([
        'message' => 'Login successful (2FA not enabled)',
        'user' => [
            'username' => $user['username'],
            'id' => $user['id']
        ],
        'totp_enabled' => false,
        'suggestion' => 'Consider enabling 2FA for better security'
    ]);
}
```

- Validation: totp_code is available and in a valid format.

```
// 2FA is enabled - require TOTP code
if (!$totp_code) {
    sendJsonResponse([
        'error' => '2FA code required',
        'message' => 'Please provide the 6-digit code from your authenticator app'
    ], 400);
}
// Validate TOTP code format
if (!preg_match('/^\d{6}$/', $totp_code)) {
    sendJsonResponse(['error' => 'TOTP code must be 6 digits'], 400);
}
```

## Response

- Decode secret and verify TOTP code

```php
// Decode secret
$secret = base64_decode($user['totp_secret']);

// Verify TOTP code
if (SimpleTOTP::verifyCode($totp_code, $secret)) {
    // Successful 2FA login
    sendJsonResponse([
        'message' => 'Login successful with 2FA!',
        'user' => [
            'username' => $user['username'],
            'id' => $user['id']
        ],
        'totp_enabled' => true,
        'login_time' => date('Y-m-d H:i:s'),
        'security_level' => 'high'
    ]);
} else {
    sendJsonResponse([
        'error' => 'Invalid 2FA code',
        'hint' => 'Check your authenticator app and ensure device time is correct'
    ], 401);
}
```

## verify.php

- 2FA SETUP VERIFICATION ENDPOINT
  - POST /verify.php
  - Body: {"username": "john", "code": "123456"}

## Request and Response

### Request

- Get user information and validate code format (6 digit)

```
$input = json_decode(file_get_contents('php://input'), true);
$username = $input['username'];
$code = $input['code'];

// Validate code format
if (!preg_match('/^\d{6}$/', $code)) {
    sendJsonResponse(['error' => 'Code must be 6 digits'], 400);
}
```

### Response

- Decode the secret.
- Verify the code and update (setup) users if verification passes.

```
$secret = base64_decode($user['totp_secret']);

// Verify the code
if (SimpleTOTP::verifyCode($code, $secret)) {
    // Enable 2FA for this user
    updateUser($username, ['totp_enabled' => true]);
    sendJsonResponse([
        'message' => '2FA setup completed successfully!',
        'status' => 'enabled', 'next_step' => 'You can now login with 2FA'
    ]);
} else {
    sendJsonResponse([
        'error' => 'Invalid code. Please check your authenticator app.',
        'hint' => 'Make sure your device time is synchronized'
    ], 400);
}
```